Claims 8-10 and 18-20 are objected to as being dependent upon a rejected base claim. However, as explained below, the claims upon which claims 8-10 and 18-20 depend are allowable. Therefore, withdrawal of these objections is respectfully requested.

Claims 1, 3, 6-7 and 21-23 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,311,591, issued to Edison M. Fischer, on May 10, 1994 *("Fischer")*. These rejections are respectfully traversed.

Claims 1, 22-13 and 16-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Fischer*. These rejections are respectfully traversed.

Claims 4-5, and 14-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Fischer* in view of U.S. Patent No. 5,758,153 issued to Bryan P. Atsatt, on May 26, 1998 *("Atsatt")*. These rejections are respectfully traversed.

CITED ART

Fischer describes a system for controlling access to files by programs. Programs are associated with a data structure referred to as program authorization information (PAI). A PAI specifies authorizations, including authorizations to access a file. (Column 5, lines 1-5).

Programs are associated with a PAI in a variety of ways. First, a program name may be mapped to a PAI. (Column 7, lines 12-40, Fig. 3A). Second, a PAI may be embedded in the program (Column 7, lines 41-46). Third, the program may be mapped to one or more PAIs associated with a particular user (Column 8, lines 43-64). "The user ... associates, for example, through an operator prompting, menu driven system, a PAI with every program to be run on a system on ..." (Column 9, lines 47-50).

When a program is called, a PCB (Program Control Block) is created for the program. At that point, a PAI for the program may be created, or if it already exists, may be modified

based on the PAIs of the program that called it. The PAI for the program is then associated with the PCB, and stored in an area of storage that cannot generally be modified by the program. (Column 17, lines 37-58).

When the program requests some action, the PAI for that program is inspected to determine whether the program is authorized to perform that action. (Column 19, lines 14-47).

Atsatt describes an object oriented framework for a file system. Specifically, *Atsatt* describes classes used to define files, directories, and users (Column 11, lines 17-27, Column 16, line 1, Column 18, line 14, Column 12, lines 25-29, Column 25, lines 29-57). Access to files in the file system is controlled through the use of protection domains. The protection domains are associated with users.

> Most operating systems provide a further level of protection for file system entities called protection domains. A protection domain is defined as a set of {Object, Rights} pairs. For a file system, the objects are the file system entities and the rights are read, write, execute, etc. An example of a file system protection domain is shown in FIG. 3B. Domain1 10 has read access to File1 and read/write access to File2, Domain2 12 has read/execute access to File3 and read access to File4, Domain3 14 has read access to File4 and read/write access to File5.

> In a file system, the protection domains are represented as a user or a group. (Copy from Column nine, lines 40-49).

The authorization to perform a particular action (e.g., right to access a file) is herein referred to as a permission. Thus, a set of permissions is a protection domain.

When a user logs in and is authenticated, an object is created to represent the user. The object is an instantiation of the class TCredentials. Before the user is allowed to access a file, a look up is performed on an ACL file. (Column 12, lines 21-40). The ACL file associates principles, i.e. users, with permissions. (Column 12, lines 33-43; Column 25, lines 58-65; Column 26, lines 10-20).

Thus, the ACL associates protection domains with objects, or in other words, with instances of a class.

CLAIM 1

Claim 1 was rejected under 35 U.S.C. 102(b) as anticipated by *Fischer*. This rejection is respectfully traversed on the grounds that at least the following limitation of Claim 1 is entirely missing from *Fischer*:

> in response to detecting the request, determining whether said action is authorized based on **permissions associated with a plurality of routines in a calling hierarchy** associated with said principal (emphasis added)

In contrast to the express requirements of Claim 1, *Fischer* does not take into account permissions of any program other than the requesting program when determining whether a requested act may be performed. Specifically, *Fischer* states that, in response to the request for an act, "an examination is made of the PAI information stored in the process control block" of the program making the request. (Column 19, lines 14-20) If that PAI does not allow access, then an error is generated (Column 19, lines 26-31). If that PAI does allow access and the request is consistent with any conventional security tests, then the act is performed. Thus, in *Fischer*, **the permissions embodied in the PAI of one and only one program (the requesting program) are taken into account when determining whether an act can be performed.** Thus, *Fischer's* system does not disclose or in any way suggest a system in which permissions associated with "a plurality of routines in a calling hierarchy" are used to determine whether an act is authorized. How *Fischer's* PAIs are initially constructed is not relevant to Claim 1, since Claim 1 is not about how permissions are established for entities, but rather how to determine whether a requested act is authorized.

CLAIMS 2-4 AND 6-10

Claims 2-4 and 6-10 depend on Claim 1, and contain all the limitations of Claim 1.

Therefore, Claims 2-4 and 6-10 are patentable for at least those reasons given with respect to

Claim 1. In addition, these claims contain limitations that independently render them

patentable over the art of record. For example, Claim 7 recites:

> determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy between and including said first routine and a second routine in said calling hierarchy

This limitation is not shown or in any way rendered obvious by the art of record. As

explained above, *Fischer* may inspect the PAIs of many programs to create a "combined"

PAI for a program, but *Fischer* never inspects the PAIs of any program but a requesting

program when determining whether a request is authorized. Consequently, *Fischer* does not

disclose or in any way render obvious a technique that takes into the account the permissions

of all routines in a calling hierarchy that are between a "privileged" routine and the routine

that actually performs the requested act.

CLAIM 5

Among other things, Claim 5 recites:

> wherein each routine of said plurality of routines is associated with a class; and wherein said association between permissions and said plurality of routines is based on a second **association between classes and protection domains** (emphasis added)

Fischer teaches use of program authorization information that is associated with the

name of the program, or that is embedded in the program. However, it does not disclose use

of program authorization information, or permissions, that are associated with a class that

define a routine. In fact, *Fischer* does not disclose or any way suggest such use of classes, or

50435-018 *(P2244/TJC)*

any mechanism for determining what classes may be associated with a program. Therefore, *Fischer* cannot disclose or suggest, in any way, "an association between classes and protection domains".

Atsatt teaches about protection domains that are associated with objects, which are instances of a class. However, *Atsatt* does not describe a system that associates a class with a protection domain.

In fact, a workable security system based on the association between a protection domain and a class would not be possible under the system described by *Atsatt*. In *Atsatt,* users are represented by objects of a single class, TCredentials. If protection domains were associated with the class TCredentials, rather than an instances of TCredentials (i.e. a user), then each user would be associated with the set of permissions associated with TCredentials. It would not be possible to associate different users with different sets of protection domains. Therefore, *Atsatt* teaches away from associating classes with protection domains. Because *Atsatt* not only fails to describe protection domains that are associated with classes, but teaches away from associating protection domains with classes, *Atsatt* fails to disclose or suggest in any way the limitations of Claim 5.

Because *Fischer* and *Atsatt*, alone or combination, fail to disclose or suggest in any way the limitations of Claim 5, it is respectfully submitted that Claim 5 is patentable. Withdrawal of the rejection with respect to and allowance of Claim 5 is respectfully requested.

Each of the remaining claims contains limitations that mirror the limitations of a claim that has already been discussed above. It is respectfully submitted that these claims are allowable for the reasons given above for the claims to which they correspond.

50435-018 *(P2244/TJC)*

For the reasons set forth above, it is respectfully submitted that all of the pending claims are now in condition for allowance. Therefore, the issuance of a formal Notice of Allowance is believed next in order, and that action is most earnestly solicited.

The Examiner is respectfully requested to contact the undersigned by telephone if it is believed that such contact would further the examination of the present application.

Please charge any shortages or credit any overages to Deposit Account No. 50-0385.

Respectfully submitted,

McDERMOTT, WILL & EMERY

Date: November 19, 1999

Brian D. Hickman
Reg. No. 35,894

(408) 271-2300
600 13th Street, N.W.
Washington, DC 20005-3096

50435-018 *(P2244/TJC)*